An Exploratory Study of Architectural Practices and Challenges in Using Agile Software Development Approaches

Muhammad Ali Babar
Lero, University of Limerick, Ireland
malibaba@lero.ie

Abstract

Agile software development approaches have recently gained popularity as a mechanism for reducing cost and increasing ability to handle change in dynamic market conditions. However, there is also a significant concern about the role and importance of the issues related to the software architecture of a system being developed using agile approaches. There is to date little empirical evidence available on what software architecture related practices are followed by teams using agile approaches and the kinds of architectural challenges resulting from using agile approaches. This paper reports a case study aimed to empirically identify and understand the architectural practices and challenges of teams using agile approaches. The findings provide useful information about the researched issues and also highlight the areas that need to be focused on for integrating agile and architecture-centric approaches.

Keywords: Software architecture, agile approaches, empirical studies, qualitative research

1. Introduction

Agile practices have recently gained popularity among a large number of companies as a mechanism for reducing cost and increasing ability to handle change in dynamic market conditions. Based on the principles of the Agile manifesto[1], researchers and practitioners have proposed several development approaches such as Extreme Programming [6], Scrum [34] and Feature-Driven Development [30]. These and other agile approaches have had significant impact on industrial software development practices. However, there is also a significant concern about the role and importance of the issues related to software architecture, which is considered one of the most important initial design artifacts. Software architecture is considered an effective tool to cut development cost and time and to increase the quality of a system [5].

Many practitioners of agile approaches appear to view software architecture in the context of the plan-driven development paradigm [28]. For them, upfront design and evaluation of software architecture requires too much work, which may have very little value to the customers of a system. Hence, they perceive architectural work as part of high ceremony processes. On the other hand, software architecture researchers and practitioners appear to believe that sound architectural practices cannot be followed using agile approaches. However, there is recently an increased recognition of the importance of paying more attention to architectural aspects in agile approaches [18, 28, 32]. Hence, there is a growing interest in identifying the mechanics and prerequisites of integrating agile and architectural approaches [23, 28].

We believe that a good understanding of the current industry practices and challenges related to software architecture is one of the most important steps towards developing appropriate strategies for integrating architectural and agile approaches. We have designed an empirical research program to identify and understand the impact of introducing agile approaches on architecture-related practices and challenges. We assert that the findings from this research program can direct future research in both architecture and agile to bridge the gap between two disciplines by developing appropriate approaches, which incorporate architecturally sounds practices in agile approaches. The findings will also help identify the strategies for improvement in organizational competencies in sound architectural practices without compromising the benefits and principles of agile development.

The objective of this paper is to disseminate the findings from an exploratory study that has gathered and analysed the qualitative data based on the opinions and experiences of practitioners who had experience in working with both plan-driven and agile approaches

[1] http://agilemanifesto.org/

978-1-4244-4985-9/09/$25.00 ©2009 IEEE

for developing large scale software intensive systems. The findings shed light on how the adoption of agile approaches impact on architecture-related practices and the kinds of architectural challenges faced by software development teams using agile approaches. As such, the objectives of this study are:

- To understand the role and importance of software architecture within software development teams using agile approaches.
- To determine the architecture-related practices and challenges of agile teams and potential solutions to deal with those challenges.

Our study has discovered several interesting findings which have enabled us to identify a few areas that need to be explored in this line of research. Our research applies an inductive approach (i.e., using facts to develop general conclusions) as an attempt to provide evidence-based insights into the interplay and tension between the requirements of architectural practices and agile principles. This paper makes three significant contributions to the increasing efforts to bridge the gap between architecture-centric and agile approaches:

- It presents the design and results of an empirical study aimed at exploring architectural practices and challenges within teams using agile approaches.
- It provides empirically founded information about how agile teams approach architectural aspects.
- It identifies the challenges and potential solutions of dealing with problems caused by architecture-related issues while using agile approaches.

2. Background and Motivation

Agile methods such as Extreme Programming (XP) [6], Crystal Clear [11] and Scrum [34] are pragmatic and minimal methodologies that are aimed to and work well in situations where the core assumptions [38] underlying agile processes hold, for example, in developing new code in the ideal context [10, 21] of agile projects where architectural design issues are not very important. Agile methods are a move away from formalism and rigor, away from centralized architecture toward a piecemeal growth of software and its architecture under a collaborative and communicative software development process [27]. Without any systematic process for the growth and changes of the architecture, the piecemeal growth of software architecture has many large uncertainties from the point of view of correctness and often results in deteriorated architectures [12].

Thapparambil [37] writes that "no agile methods discuss Architecture in any length." It has been claimed that the agile approaches consider architectural design not a very important activity. That is why the literature on the agile methods do not provide a significant amount of details or guidance on architecture design related activities [15] such as architectural analysis, architectural synthesis and architectural evaluation, as well as the artifacts associated with these activities. The agile methods tend to assume that architectural design is high-level design without explicit structuring forces such as quality attributes. Thapparambil [37] claims that "Refactoring is the primary method to develop Architecture in the Agile world." The Incremental Design primary practice of the second edition of the XP book [6] claims that architecture can emerge in daily design. The emergent design means that the architecture depends upon identifying suspicious architectural solutions in the implemented code and improving those architectural solutions when needed. According to this approach, the architecture emerges from the system rather than being imposed by some direct structuring force.

Another well known agile method Scrum's main emphasis on the architecture related practices is through face-to-face meetings and informal communication. Apart from verbal discussions related to the design decisions and overall architecture, Scrum does not place any emphasis despite the claims made about its scalability for developing and evolving large and complex systems. According to Scrum, the architecture of one-project application can always be re-factored and repackaged for a higher level of reuse after the release to production is to implement a walking skeleton, a small end-to-end functionality of the system, at the beginning of a project. The skeleton links together the main architectural components of a system. In the Incremental Re-architecture strategy of Crystal Clear [11], the team starts from a working architectural skeleton and incrementally evolves the architecture or infrastructure in stages and in parallel with the system functionality. Two architectural work products are almost certainly needed to be produced within a Crystal Clear project: system architecture and common domain model.

There are some interesting proposals for combining the strengths of the core elements of agile and architecture-centric approaches. For example, [3, 17] combine the strengths of the core elements of the risk-driven, architecture-centric Rational Unified Process (RUP) and the XP [6] process. The combinations were enabled by the facts that RUP and XP share the principles of the iterative, incremental and evolutionary development [24] and that most of the core elements of RUP and XP are complementary.

Nord and Tomayko [28] propose an integration of specific SEI architecture-centric methods into the XP framework [7]. Recently, other researchers have also been emphasizing the importance of finding a middle ground between two extreme views of architecture-centric and agile approaches [1, 9, 22].

We argue that one of the key tasks for bridging the gap between agile and architectural approaches is to identify and understand the architectural practices and challenges of teams using agile approaches. However, there is little empirical research that studies the challenges involved and industrial practices aimed to bridge the gap between the requirements of applying architecture-centric approaches and principles of agile software development methods. The main goal of our research is to help empirically identify the mechanics, challenges, and prerequisites of integrating sound architectural principles and agile approaches for developing and evolving large scale systems.

3. Research Methodology and Procedure

This section describes the research methodology and procedures used for the reported research.

We used case study method for this research. Case study is a research method that is valuable in situations where a researcher purports to understand phenomena [8] in the complex, real life context [39]. The case study is considered to be useful especially in the situations in which the context and actors of the organization are critical for the implemented study [8]. Typically, the selection of the case to be studied is done using some specific context factors [13]. The case study research was designed, implemented, and reported based on Yin's [39] steps for conducting case study research. The case of our study was selected because the studied company had been experimenting with agile approaches for the last five years and has been transitioning from planned-driven methodology to agile software development.

The primary operation of DSoft (i.e., Ficticious name) involves the provision of financial services. The company's site studied for this research had around 300 people involved in software development operations. The software products developed are supplied mainly to internal customers. Many projects involve coordinating with several teams in different parts of the World. In many cases, the requirements are generated by clients who outsource the software development to the studied site. The software development is divided between clients and vendors (i.e., the studied company) The software development in this company has been carried out following plan driven software development methods. However, the company has started experimenting with agile development methods for the last few years. The agile approaches used are mainly derivatives of Scrum [34].

We selected this particular company for the study because our colleagues were involved in collaboration with the company through a European project. This ongoing collaboration on various aspects of adopting agile approaches in this company enabled our colleagues to help us assess the suitability of this company for the reported study and gather and analyze the relevant data. The findings reported in this paper are based on the analysis of the data mainly gathered using focus group sessions [25] with software architects and project managers and semi-structured interviews with the technical leads of three different projects using agile development approaches.

Our first approach to gathering data was focus group, which is considered a proven way of exploring the perceptions, views, and experiences of a group of selected people on a defined area of interest [33]. Our focus group intended to gain a deep understanding of the architecture-related practices and challenges when using agile approaches based on the experiences and views of the participants. The discussion in focus group is largely free-flowing, but discreetly guided by a moderator, who is responsible for keeping the group discussion focused on relevant topics and make sure that everyone has an opportunity to participate [31].

We also sought data from technical leads through semi-structured interviews as one of our colleagues was to conduct interviews with technical leads on agile methods. We decided to leverage this opportunity by incorporating the questions related to architectural practices and challenges into our colleague's original questionnaire. For these interviews, a limited number of the same open ended questions used to steer the discussion in the focus group sessions were used again.

Both of our data gathering approaches (i.e., Focus group and Interviews) have known weaknesses such as subjective self-reported data based on personal opinion and interpretation of a particular event/situation, biased moderation, and small sample size, which makes generalizing the results difficult [20, 35]. To combat this, we followed a number of practices as recommended in various books on qualitative research methods [36]. These include structured questions, allocated time for each participant, transcription of sessions, and external checks of the coding labels used by the research team.

Our study design intended to organize separate focus group sessions for the technical (i.e., software architects) and management (i.e., project or team managers) people. We ran two focus groups, one for

software architects and another for project managers. There were four software architects in the first focus group session and three managers in the second focus group session. Each session started with a brief introduction of the participants and researchers. The first session lasted approximately one and half hour; the second session was one hour long. The sessions were audio recorded with participants' consent. The interviews were conducted separately.

The data analysis step involved transcribing the recorded data. The audio recording for the focus group sessions and interviews were transcribed by a professional company. One researcher read the transcription and listened to the recording for errors. This data verification step helped us to identify certain errors in the transcription. These errors were corrected. The transcribed and other data were analyzed using content analysis, a well known technique for analyzing qualitative data [35]. However, the data were not analyzed with respective to any particular coding scheme at this stage as that kind of analysis will be carried out in future after gathering data from more companies and practitioners.

4. Results and Discussion

We present and discuss the results and their implications for researchers and practitioners interested in understanding the role and importance of sound architectural practices for developing software using agile development approaches.

4.1 Demographics

The average experience of the participants of the focus group was eight years. The interviewees were playing the role of technical leads in their teams with an average experience of 5 year in developing software. All the participants for this study were purposefully selected to maximize diversity of experiences. All participants had experience of both agile and plan-driven software development. This selection was intentional, to allow informed reflection and comparison between architecture in agile and architectural issues in non-agile environments. Some had worked on projects as short as six months while others had spent four years on the same project. All of the participants also had experience of working in distributed and non-distributed development.

This research has also identified the use of various agile practices in the studied company. It was found that the company was not using a whole range of agile practices described in books like Extreme Programming [6] and Scrum [34]. Some practices were being used regularly that means a majority of the projects were using those practices such as *Sprint, Sprint Planning, Spring Review, Daily Meetings, Re-factoring, Simple Design, Coding Standards, and Collective Code Ownership*. There were certain practices that were used whenever a project team decided to use those practices such as *Testing, Post Game Sessions, and 40 Hours Week*. It was also revealed that certain agile practices (such as *Pair Programming and Onsite Customers*) were not used at all. Based on the analysis of the gathered data and knowledge of agile and architecture literature, we also identified the agile practices directly related to architectural practices and challenges. These practices are *Spring Planning, Sprint Review, Re-factoring, Metaphor, and Simple Design*. We will also discuss the relations between these agile practices and architectural practices and challenges later on.

4.2 Architecture-related Practices

In this section, we discuss the findings from the analysis of the participants' responses to the questions about architectural practices when using agile approaches. We present and discuss our findings with respect to the activities from a general design model (i.e., architectural analysis, architectural synthesis, and architectural evaluation) reported in [15], and architectural documentation and sharing of design decision and design knowledge. Before presenting and discussing our findings, we provide a brief background and description about the method of architecture design used for explaining the findings from the architectural practices in the studied projects.

Software architecture community has proposed several design methods and models such as Attribute-Driven Design (ADD) Method [5], Business Architecture Process and Organization (BAPO) [4], the Rationale Unified Process, and Siemens' 4 Views (S4V) [16]. Recently, some researchers have proposed a general model of architecture design based on five previous design models [14]. We provide a brief description of each of the three activities of this model:

- Architectural analysis – This activity aims to define the problems to be solved. An architect examines architectural concerns and context in order to come up with a set of architecturally significant requirements.
- Architectural synthesis – During this activity an architect designs architecture solutions for a set of architecturally significant requirements. An architect may consider

several available design options before selecting the ones that appears to be the most appropriate and optimal ones.

- Architectural evaluation – This activity intends to ensure that the architectural solutions chosen during the previous process are the right ones. Hence, the proposed architectural solutions are evaluated against the architecturally significant requirements.

Role of architect: We found that there had been several changes in the role and responsibilities of software architects since the introduction of agile approaches. Software architects for projects using agile approaches usually resided at the client side (i.e., offshore) and interacted with customers for getting and prioritizing User Stories, and designing Software Architectural Overall Plan (SAOP). The projects using agile approaches had a role called solution architect, which was assumed by previously called software architects. However, the solution architects took on more management oriented responsibilities. Some of the solution architects could also play the role of Scrum Master [34]. A new role called "Implementation architect" had also been introduced. This role was responsible for getting the User Stories implemented and providing technical mentoring to developers. Moreover, the implementation architect was also responsible for deciding about the timing and amount of re-factoring to be carried out. He/she would ensure that re-factoring does not have negative effect.

Architectural analysis: Our analysis revealed that most of the tasks related to architecture analysis phase (e.g., examining context and defining problems) had been pushed towards the clients since the introduction of agile. That means clients were responsible for drawing high level architectural roadmap based on their requirements and handover that roadmap to outsourcing centre. The clients were responsible for providing the project teams with the Users Stories for which detailed design decisions had to be made and implemented by the development teams. Most of the design decisions made by the solution and implementation architects of agile teams were based on the features to be delivered within fixed cost and fixed time. That means there was no attention paid to quality attributes. Further probes revealed that quality attributes did not get any attention because their achievement was not considered a measure of success by the management. Rather, it was the delivery of features within budget and time. It was also revealed that the plan-driven approach had a well define process and deliverables that were usually expected from the architectural analysis and architectural synthesis phases of the general model [15].

Architectural synthesis: Our investigation revealed that agile projects would apply two stages for design solution: software architects working with customers would draw a very high level architectural roadmap; then solution and implementation architects would make the potential design decisions considering the User Stories and their priorities, budget, time, existing platforms, and SAOP. They usually considered a limited number of potential solutions, which had normally been used in previous projects. Moreover, the agile teams produced significantly less number of deliverables from this phase and most of them were made available through a common Wiki. One respondent described the changes in these words: *"All my prior project experience in this company would have been using our proprietary design methodology. So again for each phase of the project there would have been a set of fine deliverables. So from an architecture perspective the main deliverer would be SAOP, which outlines what the architecture stack is and how it is going to be used."*

Architectural evaluation: We also discovered a few changes in the way architecture design used to be evaluated before the introduction of agile approaches. Traditionally, architecture evaluation used to be a formal process carried out by Architecture Review Board (ARB). In agile teams, architecture was evaluated at a very high level without going into details. Most of the time developers from different projects were invited to look into the architecture for serious flaws. The evaluation performed by developers was usually design validation rather than architecture evaluation. However, the participants did not report any major changes. One reason for this perception could be that they had adopted an informal evaluation even before adopting agile approaches. One architect described the changes in these words: "*Yes, there were a number of forums, so there were boards for the architectural reviews and also there would be TST that actually I used to run that forum. And again from the perspective of agile or from the perspective of hybrid, there is not a great deal of different theories except we wouldn't have gone down the TST level"*

Architecture evaluation is usually carried out for quality attributes [5]. Proponents of agile approaches claim that re-factoring can help achieve quality attributes. We explored the participants' experience about achieving quality attributes through re-factoring. We found interesting dissimilarities between the responses of the software architects and lead

developers. Software architects were of the opinion that re-factoring, both at the code level and architecture level help achieve the desired quality attributes to a certain extent such as improving maintainability by fixing the structure. However, the lead developers were of the opinion that re-factoring was much better means of fixing quality attributes problems rather than upfront design. One of them opined about re-factoring in these words: *"It is probably more valid, I think, in that, you know, rather than working off a document that might not be completely up to date, or that things may have changed in the meantime, and you need to do things differently anyway. So it is preferable to go ahead and implement it, but the down side is, that you need to be given time in your iteration, or you need to allocate time to do that. So we actually need to allocate time in the iteration to do design, then it works well, I think"*

Another interesting finding was that all the projects were concerned about the functionality, budget, and delivery dates. The achievement of quality attributes was considered as part of the maintenance projects. The participants described that maintenance projects had much larger budgets and longer life; for example, a six months development project might have several months of maintenance before being fully released with the desired quality requirements achieved.

Architecture documentation: Agile approaches advocate "*Working software over comprehensive documentation*." We found that several changes occurred in software architecture documentation practices as a result of using agile approaches. Our analysis of the participants' responses revealed that the company used to formally and comprehensively document high level architecture as well as detailed design. The SAOP used to capture the technical roadmap for a project and architecturally influential decisions. Since the introduction of agile approaches, there had been drastic reduction in the amount and details of architectural documentation. The only document that had been kept from the plan-driven practice was SAOP, which was placed on each project's Wiki. Other design decisions were also described on the Wiki. One of the respondents described the current form of architectural documentation in these words: "*Standard component diagrams and some texts around design decisions, but there isn't that much of documentation that we used to have previously. if we were following the standard process, we would have spent four weeks on a formal design specification. So, that's the difference.*"

We also explored the perceptions of the participants about the advantages or disadvantages of formally documenting software architectures. Most of the participants were of the view that the formal documentation did not add much value as one of the quantifiable benefits of adopting agile they had observed was 30% to 40% reduction in resources required for documentation. One of the participants described the advantages as follows: *"I guess the advantage is that we manage to skip a whole month of activities and just get straight into development. Other advantage was that we didn't do the usual sort of argument around doing a formal design document, and then just doing something completely different."*

Table 2 summarizes the architecture-related artifacts used by agile teams for devising and implementing design solutions.

Table 2: Presentation of the artifacts used by the agile teams in the context of the grid presented in [15].

Activities from [15]	Artifacts from [15]	Artifacts used by the agile teams
Architectural analysis	Context Requirements, Architecturally significant requirements	Platforms, fixed cost, fixed duration User stories focusing on features to be delivered No particular focus on quality attributes
Architectural synthesis	Candidate architectural solutions	Limited number of solutions known by the architecture team
	Architectural design (views, perspectives, prototypes).	Architectural Infrastructure plan (AIP) and TST documents mandated by company policy, User Stories
	Rationale	Rationale
Architectural evaluation	Quality attributes Architectural assessment	Focus on features described by User Stories No particular focus on quality attributes Inter team cooperation for design assessment
Overall process driver	Backlog	Product backlogs and Sprint backlogs

Communicating design decisions: One of the key architecture-related practices is communicating design decisions and rationale underpinning them to all relevant stakeholders [2]. The participants were asked

to describe how the architecture-related design decisions were communicated. We found that the agile teams used Wiki and design meetings for sharing design decisions. They would also use Wiki for similar objective during the plan-driven process. However, agile teams used Wiki more frequently and intensively without any formal templates or structure. Design decisions were also shared and explained on teams' whiteboards, which kept the design until it got implemented. Wikis were also used for communicating design decisions and their rationale to customers and maintenance team. The customers and maintenance teams usually got the Wiki with the final release of the software. One of the respondents described the benefits of using Wiki for capturing and communicating design decisions in these words: *"I think the whole wiki idea works pretty well. Our customers actually use a kind of waterfall methodology. They have two levels of SAOP. A very heavy duty SAOP and they also do what is called individual IP that outlines the decisions, which probably will be very similar to what we put on Wiki."*

4.3 Architecture-related Challenges

In this section, we describe some of the issues that can potentially have negative impact on architecture-related practices, artifacts or design decisions. The participants were asked about the architecture-related difficulties they experienced when using agile approaches. The participants were also asked about the currently used or proposed strategies to deal with those challenges. Our content analysis has identified the following key challenges and strategies.

Incorrect prioritization of User Stories: We found that one of the key architecture-related challenges in agile teams was that User Stories were usually prioritized without taking the technical considerations into account. It was revealed that if a critical interdependency among User Stories was found, it might have required significant re-factoring with consequences for the whole structure of the software. One of the architects described this issue in these words: *"When the User Stories come to us by then the clients have already decided about their prioritization and we have to figure out how to implement low priority User Stories before implementing the high priority ones. Such requirements usually have negative effect on design decisions because the prioritization of User Stories is not performed considering the interdependency between design decisions that need to be made in order to implement the User Stories"*

The participants proposed that the architects and developers be involved in prioritizing User Stories. The studied company had implemented this strategy by organizing "Feature Analysis Workshops" before the beginning of every project. In this workshop, all the main members of the project participated for understanding and prioritizing User Stories.

Lack of time and motivation to consider design choices: We found that agile teams were forced to focus on a limited number of solutions to achieve the required features within time and budget. One risk of this approach was architects might have missed out on a possibly better design choice by not really doing full design upfront. Moreover, developers also found themselves in the same dilemma. They were neither given upfront design nor time to come up with proper design. We know that in agile developers need to justify everything they do, so they may be forced to skip the considerations for alternative designs and implement whatever is known solution. One participant described this challenge as follows:

"That's the key difference, I think, to answer the truth, as part of each iteration, you have to give, kind of, considerations to the fact that you need to do some upfront design in each iteration. So some of the, like, some of the details, say, that you might have, if you're using the planned-driven process, you have a document that somebody has thought about a lot of design decisions upfront, and they're there on a paper. So there's less thought involved for like, say, who's doing development because, a lot of time they'll take the document, and basically implement, doing it in an agile manner, you actually have to go and think this stuff out yourself and decide on it, and agree on it, and, whatever, verify it"

One participant suggested that one way of dealing with this challenge is to have an iteration for doing architecturally focused work. Such iteration is also known as the Zero iteration among agile followers. It was proposed that such iteration can be combined with the "*Feature Analysis Workshop (FAW)*". It was also proposed that there be time allocated in each iteration for developers to think about different design choices in order to identify the most appropriate design choice.

Unknown domain and untried solutions: The participants were of the opinion that agile approaches might not be suitable when working in unknown domain, with new client, or with untried solutions. A project manager described this risk in these words: "*Working with a new solution for a new business unit is very risky if using Agile approaches. Since you would not have domain knowledge and the proven architectural solutions and patterns for that particular client's domain, it would be very hard to start delivering the features from the first iteration."*

The participants suggested that in these circumstances a hybrid approach be applied rather than pure agile one. Rather the agile approaches should be used where clients and solutions are well understood.

Lack of focus on quality attributes: Our study found that lack of focus on quality attributes for making design decisions usually results in architectural structures that can hardly meet quality requirements later on. Such systems need huge budget and time for fixing quality attributes during a maintenance project. The studied teams revealed that satisfaction of quality attributes was not a measure of success that was why they did not pay any attention to quality attributes.

The strategy to deal with this situation can be to make the satisfaction of quality attributes a measure of success and linking the budgets for development and maintenance projects. That means the work performed for quality attributes during the maintenance project should be carried out during the development project.

Other architecture-related challenges discovered by our analysis are: 1) lack of skill set is a major challenges as agile is more suitable to really good developers who know the system very well and are able to craft sophisticated design while implementing User stories without having an upfront design activity; 2) Ad hoc and unplanned documentation of design decisions on Wiki usually result in several difficulties in finding the required information about the key design decisions. Such search can consume time.

Table 3 summarizes the advantages and disadvantages of using agile related to architectural aspects of a system.

Table 3: A summary of the key advantages and disadvantage of using agile approaches.

Advantages	Disadvantages
• Bringing developers early in the picture for project design decisions • No need for spending huge amount of time on discussing and documenting solutions that may not be implemented • Clear and agreed upon deliverables for known delivery date and budget broken down into small iterations • Saving up to 30-40% resources on architectural and design documentation activities • Easily and quickly sharing design decisions and knowledge through Wikis and design meetings	• Implementing User Stories without a good knowledge of subsequent inter-dependencies of design decisions • Architecturally very risky for new projects when potential solutions are not very well understood • No time for careful design during iterations • No considerations for alterative, potentially better design choices can be missed • No focus on quality attributes except some implicit focus on performance issues • Design knowledge remains with the individuals • Searching design decisions on Wiki may be difficult

5. Limitations

All of the participants had worked with both plan-driven and agile approaches. However, the generalizability may be limited as the study was conducted in one company. But we hope that a reader may be able to identify experiences and practices that are transferable to his/her environment. There were only 10 participants in this study that can be a small sample. However, the in-depth probing, through focus groups or interviews, and content analysis can produced reliable results with small sample size. We believe that the findings are based on the experiences of those who can represent both the technical and managerial groups in the company and are decisions makers for technological and process issues.

We used a mixed methods approach as data were gathered through focus groups and interviews. We did not find any significant discrepancy in the opinions of the two groups about the architectural practices and challenges in their projects. Despite these and potentially other limitations, the empirical findings from this study are expected to provide useful information that can help understand architecture-related changes and challenges involved in using agile approaches. Moreover, this study has also found a few strategies to deal with identified challenges.

6. Conclusion and Future Work

The overall objective of our research is to empirically study and understand the principles and practices of agile software development that can influence architecture-related principles, practices, and artifacts and identify the strategies to deal with any potential negative impact. We assert that such an understanding will enable researchers and practitioners to successfully integrate architecture-centric methods and agile approaches to support the development and evolution of large scale software intensive systems. To achieve that objective, we have designed a research program consisting of a set of empirical studies aimed

at exploring practitioners' experiences and perceptions to identify and understand the changes required in architectural practices as a results of introducing agile approaches and the resulting challenges. Furthermore, our research is also aimed to find out the strategies practitioners apply (or propose) to deal with the architecture-related challenges caused by the adoption of agile approaches. This paper reports one of the studies of our research program.

This study has discovered the architectural practices and challenges of teams using agile approaches. The findings provide evidence to support the reports that the adoption of agile approaches can cause several changes in architectural practices which may have negative impact on the architectural artifacts and design decisions. It has identified the agile practices that are perceived to have significant influence on architectural practices and artifacts by the participants of this study. The findings have revealed that there are many similarities between the experiences and perceptions of experienced researchers about agile practices [23, 28] and the participants of this study.

The results provide information that can be useful for practitioners' understanding of agile approaches that can influence architectural principles and practices. Practitioners can take into account the potential effects on architectural aspects of software development while considering the introduction of agile approaches in their organizations. To deal with the challenges caused by the adoption of agile approaches and principles, practitioners can make use of not only the approaches reported by the participants of this study, but can also benefit from adapting/tailoring architecture-centric methods, techniques, and tools to support architectural principles and practices in their agile software development approaches. For example, practitioners of agile approaches can derive lightweight architecture evaluation practices from the experiences of researchers and practitioners reported in [19, 26, 29].

The research results presented here can be used by researchers in several ways. For example, the results can provide useful information for designing new research or replicating this research in order to build an empirically founded body of knowledge about the positive as well as negative effects of introducing agile practices on architecture-related practices and artifacts. For the identified architecture-related challenges caused by the introduction of agile practices, studies should be conducted to ascertain aspects of causality, in particular, what agile practices can impact different aspects of the software architecture of a system and should be considered while introducing agile practices for systems for which sound architectural practices are likely to be vital. Moreover, studies should also be conducted to determine if significant peculiarities exist for architectural practices within teams using agile approaches because of size, criticality, life, and nature of the system to be developed within one company or in different companies. The results are also expected to stimulate researchers to discover the underlying reasons and causes that can lead to the understanding and use of different architectural practices and challenges within teams using agile approaches.

We are further analyzing the data for identifying the similarities and differences among the practices and challenges reported by participants based on their role in the projects (i.e., software architects, project managers, and technical leads) and the geographically distribution of their software development teams. We hope this work will help us to relate the architectural practices and challenges with the contextual factors of projects using agile approaches. The contextualization of architectural practices and challenges is expected to enable practitioners and researchers to gain an understanding of the mechanics and prerequisites of designing and deploying suitable approaches to dealing with the architectural challenges of software development teams using agile approaches.

Acknowledgements

Author is thankful to the participants of this study. I also greatly appreciate the generosity of my colleagues Minna Pikkarainen and Kieran Conboy for helping collect the data, sharing the information they had gathered for their research on agile practices within the same projects in the studied company, and providing useful inputs for this paper. Thanks to Tuomas Iheme for providing some material in Section 2. Lero is funded by Science Foundation Ireland under grant number 03/CE2/I303-1.

7. References

[1] Ali-Babar, M. and Abrahamsson, P., Architecture-Centric Methods and Agile Approaches, *Proceedings of the 9th International Conference on Agile Processes in Software Engineering and Extreme Programming (XP08)*, 2008.

[2] Ali-Babar, M. and Gorton, I., A Tool for Managing Software Architecture Knowledge, *Proceedings of the 2nd Workshop on SHAring and Reusing architectural knowledge - Architecture, rationale, and Design Intent (SHARK/ADI 2007), Collocated with ICSE*, 2007.

[3] Ambler, S.W., Agile modeling: effective practices for eXreme Programming and the Unified Process. 2002, New York: John Wiley & Sons, Inc. 384.

[4] America, P., Rommes, E., and Obbink, H., Multi-view variation modeling for scenario analysis, in Proceedings of Fifth International Workshop on Product Family Engineering. 2003, Springer-Verlag: Siena, Italy.
[5] Bass, L., Clements, P., and Kazman, R., Software Architecture in Practice. 2 ed. 2003: Addison-Wesley.
[6] Beck, K., Extreme Programming Explained: Embrace Change. 2000: Addison Wesley Longman, Inc., Reading, MA. USA.
[7] Beck, K. and Andres, C., Extreme programming explained: Embrace change. Second ed. 2004, Reading, MA.: Addison Wesley Longman, Inc. 189.
[8] Benbasat, Goldstein, and Mead, The Case Research Strategy in Studies of Information Systems, *MIS Quartely*, 1987. **11**: pp. 369-386.
[9] Boehm, B., Get Ready for Agile Methods, with Care, *IEEE Computer*, 2002. **35**(1): pp. 64-69.
[10] Boehm, B., Get Ready For The Agile Methods, With Care, *Computer*, 2002. **35**(1): pp. 64-69.
[11] Cockburn, A., Crystal Clear: A Human-Powered Methodology for Small Teams. 2004: Addison-Wesley, Boston, USA.
[12] Coplien, J., Reevaluating the Architectural metaphor: Toward Piecemeal Growth, *IEEE Software*, 1999(September/October 1999): pp. 40-44.
[13] Eisenhardt, Building Theories from Case Study Research, *Academy of Management Review*, 1989. **14**: pp. 532-550.
[14] Hofmeister, C., et al., A general model of software architecture design derived from five industrial approaches, *Journal of System and Software*, 2006. **80**(1): pp. 106-126.
[15] Hofmeister, C., et al., A general model of software architecture design derived from five industrial approaches, *Journal of System and Software*, 2007. **80**(1): pp. 106-126.
[16] Hofmeister, C., Nord, R.L., and Soni, D., Applied Software Architecture. 2000, Reading, MA: Adison-Wesley.
[17] IBM. RUP for Extreme Programming (XP) Plug-Ins. Last accessed on Available from: http://www-128.ibm.com/developerworks/rational/library/4156.html.
[18] Ihme, T. and Abrahamsson, P., Agile Architecting: The Use of Architectural Patterns in Mobile Java Applications, *International Journal of Agile Manufacturing*, 2005. **8**(2): pp. 1-16.
[19] Kazman, R. and Bass, L., Making Architecture Reviews Work in the Real World, *IEEE Software*, 2002. **19**(1): pp. 67-73.
[20] Kontio, J., Lehtola, L., and Bragge, J., Using the Focus Group Method in Software Engineering: Obtaining Practitioner and User Experiences, *Proceedings of the International Symposium on Empirical Software Engineering*, 2004.
[21] Kruchten, P. Scaling down large projects to meet the agile sweet spot. Last accessed on Available from: http://www-128.ibm.com/developerworks/rational/library/content/RationalEdge/aug04/5558.html.
[22] Kruchten, P., Voyage in the Agile Memeplex, *ACM Queue*, 2007. **July/August**: pp. 38-44.
[23] Kruchten, P., Situated Agility, *9th International Conference on Agile Processes and eXtreme Programming in Software Engineering*, 2008.
[24] Larman, C., Agile and iterative development: a manager's guide. 2003, Boston, MA: Addison Wesley Professional. 368.
[25] Lethbridge, T.C., Sim, S.E., and Singer, J., Studying Software Engineers: Data Collection Techniques for Software Field Studies, *Empirical Software Engineering*, 2005. **10**: pp. 311-341.
[26] Maranzano, J.F., et al., Architecture Reviews: Practice and Experience, *IEEE Software*, 2005. **22**(2): pp. 34-43.
[27] Nerur, S. and Balijepally, V., Theoretical reflections on agile development methodologies, *Communications of the ACM*, 2007. **50**(6): pp. 79 - 83.
[28] Nord, R.L. and Tomayko, J.E., Software Architecture-Centric Methods and Agile Development, *IEEE Software*, 2006. **23**(2): pp. 47-53.
[29] Obbink, H., et al., Software Architecture Review and Assessment (SARA) Report, *Tech Report* SARA W.G.,, 2001.
[30] Palmer, S.R. and Felsing, J.M., A Practical Guide to Feature-Driven Development. 2002: Prentice Hall, USA.
[31] Parent, M., Gallupe, R., Salisbury, W., and Handelman, J., Knowledge Creation in Focus Groups: Can Group Technologies Help, *Information & Management*, 2000. **38**(1): pp. 47-58.
[32] Parsons, R., Architecture and Agile Methodologies - How to Get Along, in WICSA. 2008.
[33] Peffers, K. and Tuunanen, T., Planning for IS applications: a practical, information theoretical method and case study in mobile financial services *Information & Management*, 2005. **42**(4): pp. 483-501.
[34] Schwaber, K., Agile Project Management with Scrum. 2004: Microsoft Press, USA.
[35] Seaman, C.B., Qualitative methods in empirical studies of software engineering, *Software Engineering, IEEE Transactions on*, 1999. **25**(4): pp. 557-572.
[36] Strauss, A.L. and Corbin, J.M., Basics of Qualitative Research: Grounded Theory Procedures and Techniques. 1998: Sage Publications, Inc.
[37] Thapparambil, P., Agile architecture: pattern or oxymoron?, *Agile Times*, 2005. **6**(1): pp. 43-48.
[38] Turk, D., France, R., and Rumpe, B., Assumptions underlying agile software-development processes, *Journal of Database Management*, 2005. **16**(4): pp. 62-87.
[39] Yin, R.K., Case Study Research Design and Methods. 3nd ed. 2002, Thousand Oaks, CA: Sage Publications. 181.